LIMBAJE ORIENTATE PE OBIECTE

1. Moştenirea

- Reprezintă posibilitatea de a descrie clase noi (*subclase*) care preiau atât elementele ce definesc starea cît şi comportamentul (metodele) de la o altă clasă (*superclasă*).

- Implementarea clasei *stiva_spec* prin **specializare**:

```
type  stiva_spec(nr_max:integer) extends stiva=class
        operations push_2;
        procedure push_2(x,y:integer);
        begin
              push(x);
              push(y);
        end;
begin
end;
```

- Declararea unor variabile de tip clasă:

```
   var st : stiva_spec;
----------------------------------------
         st := new stiva_spec(150);
----------------------------------------
        st.push(15);
        st.push_2(155,25);
----------------------------------------
```

- Implementarea clasei *stiva_spec_spec*:

```
type stiva_spec_spec(nr_max:integer) extends stiva_spec=class
    operations under_top,top,pop;

    function top():integer;
    begin
        if not empty() then
            return tab_st[ind-1];
        else
            return –1;
        end if;
    end;

    function pop():integer;
    begin
        if not empty() then
            ind:=ind-1;
            return tab_st[ind];
        else
            return –1;
        end if;
    end;

    function under_top():integer;
    begin
        if not empty_spec() then
            return tab_st[ind-2];
        else return –1;
        end if;
    end;

    function empty_spec():boolean;
    begin
        return ind <= 2;
    end;
```

```
var st1 : stiva = new stiva(100);
    st2 : stiva_spec = new stiva_spec(50);
    st3 : stiva_spec_spec = new stiva_spec_spec(80);
    Ii: integer;
    i  := st1.under_top();          --ilegal
    i  := st2.under_top();          -- ilegal
    i  := st3.under_top();
    st1.push_2(150,12);             --ilegal
    st2.push_2(11,110);
    st3.push_2(5,17);
    i := st1.top();   --dacă stiva e goală, se generează excepție
    i := st2.top();   --dacă stiva e goală, se generează excepție
    i := st3.top();   --dacă stiva e goală se returnează  –1
```

- Avantaj major al moştenirii – posibilitatea **reutilizării**

2. Legarea dinamică

```
st : stiva;
i : integer;
cu_exceptie : boolean;
-----------------------
if cu_exceptie then
        st:=new stiva(100);
else
        st:=new stiva_spec_spec(100);
end if;
-----------------------
i:=st.top();
```

- Observații:
  - Metoda apelată în instrucțiunea *st.top* se stabileşte în mod dinamic la execuția programului, în funcție de identitatea clasei referite.
  - Variabila *st* poate referi orice obiect din clasa *stiva* sau din orice subclasă a lui *stiva*.

- Exemplu de program:

```
public class Circle {
        protected  double x, y, r;
        public static int num_circles = 0;
        public Circle(double x, double y, double r) {
            this.x=x; this.y=y; this.r=r;
            num_circles++;
        }
        public Circle(double r) {
          this(0.0, 0.0, r);
        }
        public Circle(Circle c) {
          this(c.x, c.y, c.r);
        }
        public Circle() {
          this(0.0, 0.0, 1.0);
        }
        public double circumference() {
          return 2*3.14159*r;
        }
        public double area() {
          return 3.14159*r*r;
        }
}
import java.awt.*;
public class GraphicCircle extends Circle {
        protected Color outline, fill;
        public GraphicCircle(double x, double y, double r,
                                        Color outline, Color fill);
                super(x, y, r);
                this.outline=outline;
                this.fill=fill;
        }
```

```
        public GraphicCircle(Color outline, Color fill) {
                this.outline=outline;
                this.fill=fill;
        }
        public void draw(DrawWindow dw) {
                dw.drawCircle(x,y,r,outline,fill);
        }
}
import java.awt.*;
public class GraphicCircleSmart extends GraphicCircle {
        public GraphicCircleSmart(double x, double y, double r,
                                        Color outline, Color fill) {
                super(x, y, r,outline,fill);
        }
        public GraphicCircleSmart(Color outline, Color fill){
                super(outline,fill);
        }
        public void draw(DrawWindow dw) {
                super.draw(dw);
                dw.drawLine(x-r, y, x+r, y, outline);
                dw.drawLine(x, y+r, x, y-r, outline);
        }
        public put_outline(Color outline) {
                this.outline=outline;  }
        public put_fill(Color fill) {
                this.fill=fill;  }
}
-----------------------------------------------
public class DrawWindow {
---------------------------------
        public drawCircle(double x, double y, double r,
                                        Color outline, Color fill){
        --------------------
        }
```

```
        public drawLine(double x1, double y1, double x2,
                                    double y2, Color outline){
        ----------------------   }
}
import java.awt.*
public class the_main{
        public static void main (String args[]) {
          Color forOutline=new Color( -------);
          Color forFill=new Color( -------);
          DrawWindow theFirstWd=new DrawWindow(-------);
          DrawWindow theSecondWd=new DrawWindow(-------);
          Color cl;
          double total_area=0;
          GraphicCircle tabCircle[]=new GraphicCircle[4];
          DrawWindow tabWindow[]=new DrawWindow[4];
          tabCircle[0]=new GraphicCircle(1.0, 1.0, 1.0, forOutline,
                                                  forFill);
          tabWindow[0]=theFirstWd;
          tabCircle[1]=new GraphicCircleSmart(forOutline,forFill);
          tabWindow[1]=theSecondWd;
          tabCircle[2]=new GraphicCircle(forOutline,forFill);
          tabWindow[2]=theFirstWd;
          tabCircle[3]=new GraphicCircleSmart(5.0, 7.0, 3.0,
                                                forOutline, forFill);
          tabWindow[3]=theSecondtWd;
        ---------------------------------------------------
          for(init I=0; I<tabCircle.length; I++) {
                    total_area+=tabCircle[i].area();
                    tabCircle[i].draw(tabWindow[i]); }
        ---------------------------------------------------
          cl=new Color (--------);
          tabCircle[1].put_outline(cl);    || ilegal
          ((GraphicCircleSmart)tabCircle[1]).put_online(cl);
          tabCircle[1].draw(theFirstWd);
        ---------------------------------------------------
```

Limbajul de programare C#

- Creat de Microsoft ca un instrument de dezvoltare pentru arhitectura **.**NET.

- Lansarea versiunii alfa în anul 2000, de către un colectiv de la MS condus de ANDRES HEJLSBERG.

- La fel ca şi Java, C# este şi el un limbaj derivat din C şi C++.

➢ C# a preluat de la Java unul din cele mai importante avantaje: portabilitatea programelor; aceasta se realizează pe baza *limbajului intermediar Microsoft* (MSIL) şi a componentei arhitecturii **.**NET numită *motorul comun de programare* (vezi şi cap.2).

➢ Față de Java, C# are două mari avantaje:

1. Posibilitatea de *programare în limbaj mixt*: poată să preia în *mod natural* cod scris în limbaje diferite, cerință esențială la realizarea sistemelor distribuite de dimensiuni mari.

2. *Integrarea* deplină cu platforma **Windows**.

O ierarhie simplă de clase în C#

```
using System;

// O clasa care reprezinta obiecte bidimensionale.
class Forma2D {
   public double latime;
   public double inaltime;
   public void arataDim() {
      Console.WriteLine(“Latimea si inaltimea sunt “ + latime +
                                          “ si " + inaltime);
   }
}

// clasa Triunghi deriva din clasa Forma2D.
class Triunghi : Forma2D {
   public string felTriunghi;
   public double aria() {
      return latime * inaltime / 2;
   }
   public void arataFel () {
      Console.WriteLine(“Triunghi este “ + felTriunghi);
   }
}

class Forme {
   public static void Main() {
      Triunghi t1 = new Triunghi();
      Triunghi t2 = new Triunghi();

      t1.latime = 4.0;
      t1.inaltime = 4.0;
      t1.felTriunghi = “isoscel“;

      t2.latime = 8.0;
      t2.inaltime = 12.0;
      t2.felTriunghi = “dreptunghic“;

      ConsoleWriteln(“Informatii despre t1: “);
      t1.arataFel();
      t1.arataDim() ;
      ConsoleWriteLine(“ Aria triunghiului este “ + t1.aria()) ;

      ConsoleWriteLine();

      ConsoleWriteln(“Informatii despre t2: “);
      t2.arataFel();
      t2.arataDim() ;
      ConsoleWriteLine(“ Aria triunghiului este “ + t2.aria()) ;
   }
}
```

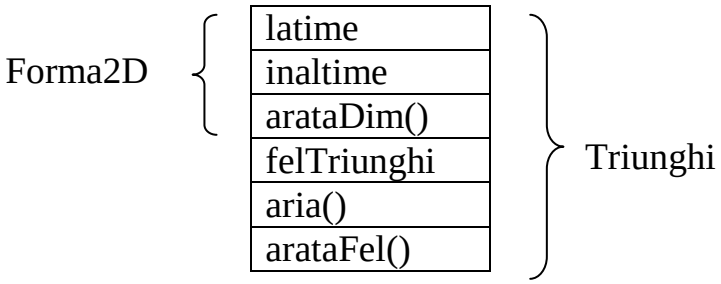
- În ceea ce priveşte moştenirea, terminologia C# păstrează aceleaşi denumiri ca în C++ (diferite de Java) şi anume:

    - superclasa se numeşte *clasă de bază*;
    - subclasa este *clasă derivată*.

- Spre deosebire de C++ dar la fel ca în Java, C# nu permite moştenirea multiplă: aceeaşi clasă derivată să moştenească mai multe clase de bază.

- Clasa `Forma2D` defineşte atributele unei forme "generice" bidimensionale care poate fi un pătrat, un dreptunghi, un triunghi etc.

- Clasa `Triunghi` este o clasă derivată a clasei `Forma2D` (moşteneşte clasa `Forma2D`) la care se adaugă câmpul `felTriunghi` şi metodele `aria`() şi `arataFel`():



- Clasa `Triunghi` poate referi câmpurile şi metodele publice ale clasei `Forma2D` la fel ca pe membrii proprii.